МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи
«Типізовані файли у Delphi»

з курсу «Програмування»
для студентів напряму 6.040302 – Інформатика
(спеціалізація «Соціальна інформатика»)

Затверджено редакційно-видавничою
радою університету,
протокол № 2 від 01.12.10.

Харків НТУ «ХПІ» 2011

Методичні вказівки до лабораторної роботи «Типізовані файли у Delphi» з курсу «Програмування» для студентів напряму 6.040302 – Інформатика (спеціалізація «Соціальна інформатика») / Уклад. М. І. Безменов. – Х. : НТУ «ХПІ», 2011. – 17 с.

Укладач М. І. Безменов

Рецензент Л. М. Любчик

Кафедра системного аналізу і управління

ВСТУП

Одним з видів файлів даних, використовуваних у програмах, написаних мовою Delphi, є типізовані файли. Використання файлів цього виду характерно, насамперед, у випадку зберігання однотипних числових даних.

Метою даної лабораторної роботи є освоєння методики визначення та обробки типізованих файлів у програмах, написаних мовою Delphi.

1. ТЕОРЕТИЧНІ ОСНОВИ

1.1. Типізовані файли та засоби роботи з ними

Крім текстових файлів у Delphi використовуються так звані типізовані файли, або файли довільного доступу, фундаментальною властивістю яких є те, що ця структура даних являє собою послідовність компонентів одного й того самого типу. Описують подібний файл словосполученням **`file of`** з наступним зазначенням типу компонентів файлу, кількість яких (довжина файлу) не фіксується:

```
var
  файлова_змінна: file of тип_компонентів;
```

Оскільки відомий тип компонентів файлу (тип файлу), а отже, й обсяг пам'яті, що відводиться під кожний з них, то можна обчислити позицію кожного з компонентів усередині файлу, що дозволяє організувати безпосередній доступ до довільного компонента типізованого файлу. Наприклад, опис

```
var
  FileInt: file of Integer;
```

свідчить про те, що компонентами файлу є дані типу `Integer`, які займають 4 байт. При цьому відпадає необхідність у спеціальному поділі окремих компонентів файлу, як це має місце в текстових файлах, і можливий довільний доступ до них (у цьому розумінні типізований файл дещо нагадує одновимірний масив).

Деякі типи даних (наприклад, тип **`string`** або динамічні масиви) не можуть бути записані у файл, оскільки під час виконання обсяг пам'яті, що відводиться під них, може змінюватися, і компілятор не може організувати дії по завершенню процесу запису таких даних у файл або читання їх з файлу. У той же час тип `ShortString` для типу компонентів файлу є допустимим.

Щоб можна було працювати з типізованим файлом, потрібно, як і у випадку текстових файлів, спочатку зв'язати ім'я файлової змінної з зовнішнім ім'ям файлу (оператор `Assign`), а потім відкрити його (оператори `Reset` та `Rewrite`, але не `Append`). Оператори `Reset` і `Rewrite` відкривають файл і для читання, і для запису (а не тільки для читання або тільки для запису, як це має місце в текстових файлах). Відмінність їх полягає у тому, що оператор `Reset` відкриває тільки існуючий файл (якщо такого файлу немає, то буде помилка часу виконання), а оператор `Rewrite` створює новий файл (якщо файл з таким ім'ям уже є, то він буде знищений і створений заново). При відкритті файлу з ним зв'язується поточний вказівник файлу, що позиціюється на його перший компонент. Оперувати можна тільки тим компонентом файлу, на який указує вказівник файлу. При читанні або записі компонента файлу відбувається автоматичне переміщення вказівника на наступний компонент. Читання з типізованого файлу проводиться оператором `Read` (але не `ReadLn`), а запис у нього – оператором `Write` (але не `WriteLn`), що збігаються за форматом з аналогічними операторами для текстових файлів; однак у списку виведення оператора `Write` можуть бути **тільки змінні**. Типи компонентів файлу і типи змінних у списках введення/виведення повинні бути сумісними за присвоюванням. Компонентами типізованих файлів можуть бути числові та символьні значення, значення логічного типу, короткі рядки, масиви, множини, записи, але не файли або структури з файловими компонентами.

Слід зазначити, що оператор `Read` здійснює формальне читання з типізованого файлу тієї кількості байт, яка визначається типом файлу, і інтерпретує вміст цих байтів відповідно цьому типу. Якщо, наприклад у файл було записано дані типу `Real`, а він відкритий як файл типу `Integer`, читання проведене буде, але при цьому зчитані дані будуть інтерпретовані неправильно (найпростіший приклад: числа `1` та `1.0000000` мають різний формат подання у пам'яті). Це стосується і такого терміна, як «цілочисловий файл» – цілочисловими є типи `Byte`, `Integer`, `Word` та ін., але формат їх подання різний (різна кількість байт, наявність або відсутність знаку). Таким чином, завжди необхідно знати не тільки те, які дані зберігаються у файлі, але і їх формат.

Тип змінних, що вказують у списку уведення оператора `Read` при читанні з типізованого файлу, повинен збігатися з типом файлу, навіть якщо тип змінної є більш загальним у порівнянні з типом файлу. Наприклад, для описів

```
var
  n: Integer;
```

```
m: LongInt;
k: Byte;
a: Real;
b: Extended;
d: Double;
f1: file of Byte;
f2: file of Integer;
f3: file of Real;
```

недопустимими будуть такі оператори:

```
Read(f1, n);
Read(f2, a);
Read(f3, b);
Read(f3, d);
```

хоч в них змінні мають поглинаючий тип відносно базового типу файлу.

Є виключення, що стосується типів `LongInt` й `Integer` – з файлу типу `Integer` можна читати дані, записуючи їх у змінні типу `LongInt`, і навпаки. Відповідно до цього для наведених вище описів оператор

```
Read(f2, m);
```

буде не тільки синтаксично правильним, але й буде виконуватися коректно.

При записі в типізований файл за допомогою оператора `Write` обмеження менш жорсткі. У цьому випадку тип змінних зі списку виведення та базовий тип файлу повинні відноситися до одного виду (обидва дійсні, обидва цілі й т. д.). Так, для наведених вище описів допустимими будуть такі оператори:

```
Write(f1, n);
Write(f2, k);
Write(f3, b);
Write(f3, d);
```

Однак оператор

```
Write(f3, n);
```

все-таки буде синтаксично неправильним.

Природно, сказане вище не стосується випадку, коли базовим типом файлу є який-небудь складений тип (масив, запис, множина). У цьому разі типи файлу і елементів списку уведення/виведення повинні бути еквівалентними.

Таким чином, можна **рекомендувати** наступне: базовий тип файлу і тип елементів списку уведення/виведення **повинні бути однаковими**.

Якщо у файл записувався вміст масиву (або декількох масивів), то його можна трактувати і як файл, компонентами якого є не масиви, а дані, тип яких збігається з базовим типом масиву. Так, якщо у файлі f.dat міститься 80 чисел, записаних у форматі даних типу `Real`, то для описів

```
type
  TArr8 = array[1..8] of Real;
  TArr10x8 = array[1..10] of TArr8;
var
  a: TArr10x8;
  i, j: Integer;
  f1: file of TArr10x8;
  f2: file of TArr8;
  f3: file of Real;
```

еквівалентними за отримуваним результатом є такі фрагменти програмного коду:

1)
```
AssignFile(f1, 'f.dat');
Reset(f1);
Read(f1, a);
```

2)
```
AssignFile(f2, 'f.dat');
Reset(f2);
for i := 1 to 10 do
  Read(f3, a[i]);
```

3)
```
AssignFile(f3, 'f.dat');
Reset(f3);
for i := 1 to 10 do
  for j := 1 to 8 do
    Read(f3,a[I, j]);
```

Дізнатись кількість компонентів типізованого файлу (розмір файлу) можна, звернувшись до функції

`FileSize(`файлова_змінна`)`.

Наприклад, якщо змінна `k` має тип `Integer`, а `f` – файлова змінна типізованого файлу, то оператор `k:=FileSize(f)` записує в змінну `k` розмір файлу `f` (функція `FileSize` повертає значення типу `Integer`).

Компоненти типізованого файлу (але не байти) нумеруються, причому починаючи з 0 (порядковий номер останнього елемента файлу на одиницю менший від розміру файлу). Щоб довідатися, на якому компоненті розташовується вказівник файлу, використовують функцію FilePos, яка повертає значення типу LongInt і як параметр має файлову змінну:

FilePos(файлова_змінна).

Положенням поточного вказівника можна керувати, для чого служить процедура Seek, що має формат

Seek(файлова_змінна, номер_компонента).

Другий параметр, що має типтип LongInt, задає номер компонента з відліком від 0, на який повинен перемістится вказівник файлу:

Seek(f, 5) – перейти до компонента файлу f, що має номер 5 (фактично шостий компонент);

Seek(f, FilePos(f) - 1) – перейти до попереднього компонента;

Seek(f, FileSize(f)) – перейти на кінець файлу.

Закриття типізованого файлу, як і для текстових файлів, здійснюється процедурою

CloseFile(файлова_змінна).

Як і для текстових файлів, для типізованих можна використовувати функцію

Eof(ім'я_файлу),

що повертає значення True, якщо поточний вказівник розташований на ознаці кінця файлу, тобто при виконанні рівності

FilePos(ім'я_файлу) = FileSize(ім'я_файлу).

Процедура Seek і функції FilePos та FileSize дозволяють легко коректувати компоненти файлу й організовувати дозаписування у кінець файлу зі збільшенням його розміру.

Процедура

Truncate(файлова_змінна)

знищує всі компоненти типізованого файлу, ім'я якого зазначене як її параметр, починаючи з компонента, на якому розташований поточний вказівник. Однак знищити компонент усередині файлу не можна, для цього файл повинен бути переписаний.

У той час як текстові файли можуть бути створені текстовим редактором, типізовані файли створюються в результаті роботи Delphi-програми.

1.2. Організація пошуку файлів

Функція

`FindFirst(`шлях, атрибути, перший_запис`)`

виконує пошук першого запису, що відповідає заданим імені (параметр шлях типу **`string`**) і набору атрибутів (атрибути, тип `Byte`).

Запис може відшукуватися не тільки за конкретним ім'ям (з можливим розширенням), але і серед файлів та каталогів, що задаються за маскою в першому параметрі. При цьому маска задається звичайним для Windows чином, а саме, у масці можна подавати символ `*`, який служить для зазначення того, що замість нього може стояти будь-яка допустима для імені файлу або розширення послідовність символів (у тому числі і нульовій довжині), а також символ `?`, що служить для позначення будь-якого допустимого символу (одного). Ім'я, що задається, може бути повним (містити ім'я диска і маршрут доступу) або скороченим. В останньому випадку діє звичайний для Windows принцип умовчання.

Набір атрибутів (ознак) – це деяке числове значення. Окремим атрибутам відповідають наступні константи модуля `SysUtils` (у коментарях зазначено зміст констант):

```
faReadOnly  = $01,              // Тільки для читання
faHidden    = $02,                  // Прихований файл
faSysFile   = $04,                   // Системний файл
faVolumeID  = $08,                       // Мітка тому
faDirectory = $10,//Ім'я директорія (каталогу, папки)
faArchive   = $20,                    // Архівний файл
faAnyFile   = $3F.                      // Усякий файл
```

Комбінації атрибутів задаються іншими числовими значеннями, які звичайно отримують за допомогою застосування бітової операції **`and`** (або інших бітових операцій) до операндів, що задаються наведеними вище значеннями атрибутів. Результати роботи функції записуються в третьому параметрі, тип якого описано у модулі `SysUtils` так:

```
type
  TSearchRec = record
```

```
    Time: Integer;
    Size: Integer;
    Attr: Integer;
    Name: TFileName;
    ExcludeAttr: Integer;
    FindHandle: THandle;
    FindData: TWin32FindData;
  end;
```

У цьому параметрі відображаються характеристики першого знайденого файлу, що зареєстрований у зазначеному каталозі: упакований час створення або останнього оновлення (поле `Time`), розмір файлу в байтах (поле `Size`), значення атрибута (поле `Attr`), ім'я й розширення файлу або каталогу (поле `Name`), додаткова інформація про час створення та останнього доступу, а також про повне і скорочене ім'я (поле `FindData`). Значення поля `Time` за допомогою функції

`FileDateToDateTime(`час_створення_файлу`)`

можна перетворити до типу `TDateTime`, у якому повертає результат своєї роботи дана функція.

Зазначимо, що зворотне перетворення виконує функція

`DateTimeToFileDate(`значення_типу_`DateTime)`,

яка повертає результат типу `Integer`, у якому задається системний час створення файлу.

Функція

`FindNext(`наступний_запис`)`

служить для пошуку на диску наступного запису з тими ж параметрами, що й у раніше виконаній функції `FindFirst` або `FindNext`. Її параметр має тип `SearchRec`.

Оскільки для роботи підпрограм `FindFirst` і `FindNext` виділяється додаткова пам'ять, її рекомендується звільняти по закінченні пошуку. Це забезпечується використанням функції

`FindClose(`запис_про_файл`)`,

параметром якої є змінна, що вживалося в раніше виконаній функції `FindNext` або `FindFirst`.

При нормальному завершенні функції `FindFirst` і `FindNext` повертають значення `0`, інакше – код помилки.

2. ПРИКЛАДИ ПРОГРАМ

Приклад 1. У текстовому файлі data.txt записано цілі числа, розділені довільною кількістю будь-яких пробільних символів, з можливими порожніми рядками в будь-якому місці файлу. Переписати в новий файл res.sol у форматі чотирибайтових цілих чисел зі знаком тільки ті з чисел, які кратні 3.

Розв'язання. Розмістимо на формі тільки кнопку Button1. В такому разі задачу розв'язує такий опрацьовувач події OnClick кнопки Button1:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  fInp: TextFile;
  fOut: file of LongInt;
  m: LongInt;
begin
  AssignFile(fInp, 'data.txt');
  Reset(fInp);     // Існуючий файл відкриваємо для читання
  AssignFile(fOut, 'res.sol');
  Rewrite(fOut);                  // Створення нового файлу
  while not SeekEof(fInp) do begin
    Read(fInp, m);
    if m mod 3 = 0 then
      Write(fOut, m);           // Запис у типізований файл
  end;
  CloseFile(fOut);                          // Закриття файлу
  CloseFile(fInp);                          // Закриття файлу
  Caption := 'Ok';
  Button1.Enabled := False;
end;
```

Приклад 2. Компонентами файлу data.dbl є дані, записані у форматі типу double. Поміняти місцями ліву і праву половини цього файлу, не використовуючи допоміжний файл. Якщо у файлі непарна кількість чисел, то число, що розміщене в середині файлу, не переміщати.

Розв'язання. Скористаємося формою з прикладу 1 з таким програмним кодом опрацьовувача події OnClick кнопки Button1:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  f: file of Double;
  pLeft,                    // Позиція у лівій половині файлу
  pRight: LongInt;         // Позиція у правій половині файлу
  dL, dR: Double;                          // Допоміжні змінні
begin
  AssignFile(f, 'data.dbl');
  Reset(f);                        // Відкриваємо існуючий файл
   // Обчислюємо першу позицію вказівника у правій половині
  pRight := FileSize(f) - FileSize(f) div 2;        // файлу
                  // Як значення параметра циклу  - позиція
                                        // вказівника праворуч
  for pLeft := 0 to FileSize(f) div 2 - 1 do begin
    Seek(f, pLeft);      // Вказівник – на компонент ліворуч
    Read(f, dL);        // Читаємо компонент у лівій частині
    Seek(f, pRight);    // Вказівник – на компонент праворуч
    Read(f, dR);       // Читаємо компонент у правій частині
    Seek(f, pRight);    // Вказівник – на компонент праворуч
    Write(f, dL);                 // Записуємо число праворуч
    Seek(f, pLeft);      // Вказівник – на компонент ліворуч
    Write(f, dR);                  // Записуємо число ліворуч
    Inc(pRight);     // Майбутня позиція вказівника праворуч
  end;
  CloseFile(f);                                // Закриття файлу
  Caption := 'Ok';
  Button1.Enabled := False;
end;
```

Приклад 3. У поточній теці знаходиться файл notebook.dat, у якому зьерігаються телефонні номери у форматі записів з двома полями – прізвище (короткий рядок довжиною 20 символів) та номер телефону (чотирибайтове ціле число). На телефонній станції шестизначні номери, що починаються з 37, замінюються семизначними номерами, що починаються з 337. Внести зміни у файл notebook.dat.

Розв'язання. Розмістимо на формі кнопку Button1 та багаторядковий редактор для контрольного виведення, наддавши йому ім'я mmOutput1. Скористаємося таким опрацьовувачем події OnClick кнопки Button1:

```
procedure TForm1.Button1Click(Sender: TObject);
type
```

```
  t_subscriber = record
    surname: string[20];
    tel: LongInt;
  end;
var
  subscriber: t_subscriber;                          //Абонент
  f: file of t_subscriber;
  s: string;
begin
  AssignFile(f, 'notebook.dat');
  Reset(f);                                        //Існуючий файл
    //Нижче файл розглядається як файл послідовного доступу
  while not Eof(f) do begin
    Read(f, subscriber);        //Читання компонента з файлу
    with subscriber do begin
      Str(tel, s);
      if (Length(s) = 6) and (Copy(s, 1, 2) = '37') then
      begin
        Insert('3', s, 1);
        tel := StrToInt(s);
      end;
    end;
                            //При читанні вказівник змістився
              //Повертаємо його назад (вже довільний доступ)
    Seek(f, FilePos(f) - 1);
                                    //Змінюємо компонент файлу
    Write(f, subscriber)
  end;
  CloseFile(f);                                  //Закриття файлу
  Reset(f);
  mmOutput1.Lines.Add('Контрольне виведення');
  while not Eof(f) do begin
    Read(f, subscriber);
    with subscriber do
      mmOutput1.Lines.Add(surname + #9 + IntToStr(tel));
  end;
  CloseFile(f);
  Caption := 'Ok';
  Button1.Enabled := False;
end;
```

Приклад 4. Переіменуватиа усі файли, що розташовані у поточній теці та мають розширення .dat, дописавши спереді до імені порядкрвий номер файлу.

Розв'язання. Розмістимо на формі тільки кнопку Button1 і скористаємося таким опрацьовувачем події OnClick кнопки:

```
rocedure TForm1.Button1Click(Sender: TObject);
var
  at: Integer;
  file_rec: TSearchRec;
  f: file  of Integer;                    // Краще f: file;
  Code, Number: Integer;
begin
  Number := 1;
      // Пошук першого файлу (каталогу, мітки тому)
             // з розширенням dat
  Code := FindFirst('*.dat', faAnyFile, file_rec);
  while Code = 0 do begin              // Поки пошук успішний
    at := file_rec.Attr;          // Запам'ятовуваємо атрибут
                       // Якщо це не каталог і не мітка тому
    if at and (faDirectory or faVolumeID) = 0 then begin
      AssignFile(f, file_rec.Name);
      Rename(f, IntToStr(Number) + file_rec.Name);
      Inc(Number);
    end;
    Code := FindNext(file_rec);    // Шукаємо наступний файл
  end;
  FindClose(file_rec);                  // Звільняємо пам'ять
  Caption := 'Ok';
  Button1.Enabled := False;
end;
```

3. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

За час, відведений для виконання лабораторної роботи (2 академічні години), студент повинен:

1. Розробити алгоритм розв'язання задачі, запропонованої для програмування.
2. Здійснити проектування форми для функціонування розроблюваної програми.
3. Здійснити програмну реалізацію розробленого алгоритму.

4. Здійснити відлагодження програми, виправивши синтаксичні та логічні помилки.
5. Підібрати тестові дані для перевірки програми, включаючи виняткові випадки.
6. Оформити звіт до лабораторної роботи.
7. Відповісти на контрольні запитання.
8. Здати викладачу працездатну програму з демонстрацією її роботи на декількох варіантах вихідних даних.

4. ВАРІАНТИ ЗАДАЧ

1. Дано файл цілих чисел. Записати в інший файл найбільше значення перших 10 компонентів, потім – наступних 10 компонентів і т. д. Якщо в останній групі менше 10 чисел, то розглядати неповну групу.
2. Дано файл із цілих невід'ємних чисел. Переписати в інший файл числа, що читаються однаково зліва направо і справа наліво.
3. Дано файл цілих чисел. Записати в новий файл парні компоненти початкового файлу у зворотному порядку.
4. У файлі f записано цілі числа. Переписати його вміст у два файли. У перший файл записати всі парні числа з непарними номерами, а в другий – решту чисел. Компоненти у нових файлах повинні йти в порядку, зворотному порядку їх розміщення у початковому файлі
5. Дано файл дісних чисел. Без використання іншого файлу переписати його вміст таким чином, щоб спочатку йшли всі додатні числа, а потім всі недодатні без зміни взаємного розташування елементів усередині кожної із цих двох груп.
6. Дано файл із дійсних чисел. Упорядкувати його вміст у наступному порядку: перше, останнє, друге, передостаннє й т. д.
7. Дано впорядкований за зростанням файл цілих чисел. Розширити цей файл, включивши в нього новий компонент зі збереженням властивості впорядкованості.
8. Дано файл f, компоненти якого є цілими числами. Записати у файл g всі прості числа, що входять у файл f. Числа у файлі g повинні йти:
   а) за неубуванням;
   б) у порядку убування без повторень.
9. Дано файл f, компоненти якого є цілими числами. Одержати файл g, утворений з файлу f виключенням повторних входжень того самого числа.

10. Дано файл цілих чисел, кількість компонентів якого кратна 4. Числа у файлі йдуть у такому порядку: два парні, два непарні, два парні, два непарні і т. д. Змінити вміст файлу таким чином, щоб порядок чисел був таким: парне, непарне, парне, непарне і т. д.
11. Дано файл цілих чисел. Записати в інший файл найбільше значення перших 10 компонентів, потім – наступних 10 компонентів і т. д. Якщо в останній групі менше 10 чисел, то розглядати неповну групу.
12. Дано файл з ненульовими дійсними компонентами, кількість яких кратна 2. Розглядаючи компоненти файлу послідовними парами, поміняти місцями елементи пар, якщо вони мають різні знаки.
13. Уводяться цілі числа до першого нуля (всі числа різні). Знайти, де сума елементів більше: до максимуму або після. Відповідний фрагмент списку введених даних зберегти у файлі, записавши дані у порядку, зворотному порядку введення.
14. Дано натуральне число k і файл цілих чисел, кількість компонентів якого кратна $4k$. Числа у файлі йдуть послідовними групами з k парних і k непарних чисел. Поміняти місцями групи таким чином, щоб числа йшли послідовними групами:
    а) з $2k$ парних і $2k$ непарних значень;
    б) з $2k$ непарних і $2k$ парних значень.
15. Дано два файли f_1, f_2 з даними одного й того самого типу. Сформувати файл f_3, що є «різницею» файлів f_1 і f_2, тобто містить усі компоненти файлу f_1, які не входять у f_2.
16. Дано два файли f_1, f_2 з даними одного й того самого типу. Сформувати файл f_3, який є перетином файлів f_1 і f_2, тобто містить усі компоненти, що входять як у f_1, так і в f_2.
17. Дано файл цілих чисел. Вивести кількість ділянок цього файлу, що складаються:
    а) з послідовно розміщених парних елементів;
    б) з послідовно розміщених однакових елементів.
18. Дано файл, компонентами якого є числа відмінні від нуля. Компоненти цього файлу записані в наступному порядку: п’ять додатних, п''ять від’ємних, п’ять додатних, п’ять від’ємних і т.д., причому кількість компонентів кратна 20. Переписати його вміст в інший файл так, щоб числа йшли в наступному порядку: десять від’ємних, десять додатних, десять від’ємних, десять додатних і т. д.

5. КОНТРОЛЬНІ ЗАПИТАННЯ

1. У чому особливість типізованих файлів? Як вони оголошуються?
2. Які засоби для роботи з типізованими файлами існують у Delphi?
3. Для чого служить поточний вказівник файлу?
4. Чи можна управляти положенням поточного вказівника файлу?
5. Як довідатися про кількість компонентів типізованого файлу?
6. Як нумеруються компоненти типізованого файлу?
7. Як дізнатися положення вказівника файлу?
8. Як співвідносяться тип файлу і тип елементів списку введення або виведення?
9. Як записати у типізований файл числову константу?
10. Чи може бути типом компонентів файлу рядковий тип?
11. Чи можна записати у типізований файл єлемент даних типу «запис»?
12. Чи можна записати у типізований файл вміст масиву, не організуючи перебирання його елементв у циклі?
13. Як можна здійснити пошук файлу?
14. Для чого служить набір атрибутів файлу?
15. Назвіть константи, що задають окремі атрибути.

СПИСОК ЛІТЕРАТУРИ

1. Безменов, М. І. Основи програмування в середовищі Delphi : навч. посіб. / М. І. Безменов. – Х. : НТУ «ХПІ», 2010. – 608 с.
2. Кэнту, М. Delphi 7 : Для профессионалов / М. Кэнту – СПб. : Питер, 2004. – 1101 с.
3. Архангельский, А. Я. Программирование в Delphi 6 / А. Я. Архангельский. – М. : БИНОМ, 2002. – 1120 с.
4. Дарахвелидзе, П. Г. Программирование в Delphi 7 / П. Г. Дарахвелидзе, Е. П. Марков. – СПб. : БХВ-Петербург, 2003. – 784 с.
5. Культин, Н. Б. Основы программирования в Delphi 7 / Н. Б. Культин. – СПб. : БХВ-Петербург, 2003. – 608 с.
6. Пестриков, В. М. Delphi на примерах / В. М. Пестриков, А. Н. Маслобоев. – СПб. : БХВ-Петербург, 2005. – 496 с.
7. Ремкеев, А. А. Курс Delphi для начинающих. Полигон нестандартных задач / А. А. Ремкеев, С. В. Федотова. – М. : СОЛОН-Пресс, 2006. – 360 с.

Навчальне видання

Методичні вказівки
до лабораторної роботи
«Типізовані файли у Delphi»
з курсу «Програмування» для студентів напряму 6.040302 – Інформатика
(спеціалізація «Соціальна інформатика»)

Укладач БЕЗМЕНОВ Микола Іванович

Відповідальний за випуск О. С. Куценко
Роботу до видання рекомендував О. В. Горелий

За авторською редакцією

План 2011 р., поз. 8 / 72-11

Підп. до друку 23.05.2011 р. Формат 60 × 84 $^{1}/_{16}$. Папір офісний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 0,9. Наклад 50 прим.
Зам. № 164. Ціна договірна.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 3657 від 24.12.2009 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ «ХПІ», 61002, Харків, вул. Фрунзе, 21